NUSC Technical Report 6423
10 February 1981

(12)

Speech Synthesis From Unrestricted Text Using A Small Dictionary

Richard Loose
Combat Control Systems Department

Naval Underwater Systems Center
Newport, Rhode Island / New London, Connecticut

Preface

This research was conducted under IR/IED Project No. A60100, "Man Machine Audio Communication in ASW Combat Control," Principal Investigator, Edward A. DeGregorio (Code 3512), Navy Subproject and Task No. ZR000-0101/61152N.

This report was submitted by the author in partial fulfillment of the requirements for the Degree of Bachelor of Science, Electrical Engineering, at the Massachusetts Institute of Technology, June 1980.

The technical reviewer for this report was Dr. John Allen, Massachusetts Institute of Technology, Cambridge, MA.

I would like to thank my supervisors at NUSC, Kevin Higgins, Ed Degregorio, and Tony Bessacini for allowing me the freedom to undertake this project. Prof. Jon Allen at MIT for supervising this thesis, the many researchers involved in the voice synthesis field, without whose work this project would not be possible, and Eric J. Swenson for his invaluable help in the printing of this thesis.

Reviewed and Approved: 10 February 1981

W A Von Winkle

W. A. Von Winkle
Associate Technical Director for Technology

Inquiries concerning this report should be referred to Code 351,
Newport Laboratory, Naval Underwater Systems Center,
Newport, Rhode Island 02840

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR 6423 ✓	2. GOVT ACCESSION NO. AD-A096	3. RECIPIENT'S CATALOG NUMBER 883
4. TITLE (and Subtitle) 6) SPEECH SYNTHESIS FROM UNRESTRICTED TEXT USING A SMALL DICTIONARY.		5. TYPE OF REPORT & PERIOD COVERED 9) Technical rept.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10) Richard/Loose 16) ZR00001		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Underwater Systems Center Newport Laboratory Newport, Rhode Island 02840	17)	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS A60100 ZR000-0101/61152N
11. CONTROLLING OFFICE NAME AND ADDRESS 14) NUSC-TR-6423	11)	12. REPORT DATE 10 Feb[redacted] 1981
		13. NUMBER OF PAGES 26 12) 28
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Present voice synthesis systems have either used a large dictionary to look up words to be voiced, or have used only letter-to-sound rules resulting in monotone speech. The system outlined in this thesis, SIMON, utilizes the morphographemic analysis aspect of the large systems, while relying on letter-to-sound rules for the translation of morpheme to morph. The result is a system which produces inflected speech without the storage requirements of a large dictionary.

40'96 .4

Table of Contents

Chapter 1

Thesis Structure

This paper will be presented in eight chapters. Each chapter's first section gives an overview of that particular aspect of speech synthesis and motivates the methods incorporated in the system. These sections, taken as a whole, are self-contained and cover the entire system although they lack a great deal of specificity provided by the remaining sections in each chapter which detail the working of the system and include explanatory examples.

There are four appendices dealing with program implementation: Implementation, Memory Management, Data Structures, and Program Structure and Data Flow. They are not intended as program documentation, but rather provide the basic scheme of implementation.

Chapter II

Word Parsing

1. Overview

In order to accomplish the parsing of phrases and the assignment of stress, the parts-of-speech of each word in the sentence must be determined. This identification is made by first checking each word against a lexicon and, if not found, performing a morphological analysis of the word. The result is the association of each word with a set of composite morphs and a part-of-speech. For example, "kindness" becomes associated with "kind" + "ness" and is marked as a noun. If neither the lexicon nor the morphological analysis can provide the part-of-speech, the word is labeled as an "unknown".

Lookup in the lexicon is straight forward. Ambiguous morphemes are listed as such; e.g., "laugh" would be listed as word that could function as a singular noun or plural verb. The resolution of these ambiguities is attempted in the phrase parsing phase of the translation.

If the word is not found directly in the lexicon, it is examined for possible suffixes. The morphological stem changing rule used are those presented in [Winograd, 1971]. The algorithm takes the suffix "s" as a special case, and handles examples such as "buzzes" -> "buzz" + "s" and "marries" -> "marry" + "s". It also accounts for morphological changes caused by vocalic suffixes; e.g., "advisable" -> "advise" + "able". It is assumed that consonantal suffixes do not produce morphological stem changes. The procedure is repeated to allow for multiple suffixes. A list of suffixes is generated upon their removal. Prefixes are taken off the resulting stem, and prepended to the list of composite morphs.

Since the stem left over after affix removal is likely not to be in the lexicon, a criterion for determining whether the affix should be stripped off is needed. The test used requires that the stem left over has at least three characters and contains a vowel.

The assumptions the algorithm makes are not always justified; hence, errors will result. For example, "argument' -> "argu" + "ment" leaving the "e" off of "argue", and, were "under" not in the lexicon, we would have "under" -> "und" + "er". It is hoped that these errors do not significantly affect the quality of the final voice output.

No attempt is made to break compound words since the lexicon is not large enough to handle this type of analysis.

2. Affix Separation

To aid in the decomposition of words into morphs, the lexicon lists morphs according to their properties, as outlined in "Man-to-machine communication by speech, Part 1: Generation of segmental phonemes for text" [F. F. Lee, 1968]. These categories are

1) free morph that never combines with others (e.g. "me")

2) free morph that may combine with others (e.g. "house")

3) vocalic suffix (e.g. "-able")

4) consonantal suffix (e.g. "-ness")

5) prefix (e.g. "pre-")

In addition to morph type, the lexicon provides the part-of-speech of free morphs and the resulting part-of-speech and allowable roots for affixes. The rules for removing affixes are embedded procedurally in the word parser.

The parser first takes off the suffix "s" or "es" if applicable, altering the resulting stem as in "An A.I. Approach to English Morphemic Analysis", [Winograd, 1971]. The algorithm then strips off suffixes by matching the last characters of the word against the list of suffixes provided by the lexicon. After each suffix is removed, the resulting stem is transformed if necessary (e.g. "making" -> "make" + "ing") and looked up in the lexicon to determine if the parse.is done. A suffix is not removed if:

1) the resulting stem has less than three characters

2) the resulting stem does.not contain a vowel

3) the resulting stem is subsequently determined to be of a type that does not allow the suffix to be added (e.g. "-tion" does not add to nouns)

The procedure is iterated until no more suffixes can be removed. Prefixes are removed in an identical fashion.

For example, "dependent" first has the suffix "-ent" removed, producing "depend" + "ment". No stem changes are necessary for consonantal suffixes. The resulting stem has no recognizable suffix, so prefixes are looked for. The pre "de-" is taken off to produce "de" + "pend" + "ent". "Dependent" is labeled as a singular noun with a morph decomposition of "de" + "pend" + "ent".

Chapter III

Letter-To-Sound Conversion

1. Overview

Morphs not listed in the lexicon receive their phonemic representation by means of letter-to-sound rules. The resulting phonemic representation tells the voice system how to pronounce words and is used to determine stress levels within the word. The rules applied are a modification of the rules presented in "Automatic Translation of English Test to Phonetics by Means of Letter-to-Sound Rules", [Elovitz, et al, 1976]. The modifications are minimal, and account for SIMON's list of pre-translated entries in the lexicon, as well as deletion of rules pertaining to affixes.

The algorithm makes a single left-to-right pass over the input stream of characters, and translates each character or group of characters to a phonemic equivalent based on the context of the characters being translated. For example, "gate" -> "g" + "e" + "t" (where the phonemes are represented by standard International Phonetic Alphabet (IPA) symbols).

2. Algorithm for Conversion

Rules are of the form: characters X, preceded by the pattern Y, and followed by the pattern Z, translates phonemic representation W, where X is one or more characters, Y and Z are patterns to be matched by the preceding and following characters, respectively, and W is a string of phonemes. A "pattern" is, e.g., "one or more consonants", or "an 'e' followed by a voiced consonant". The rules are applied character by character as a pointer scans across the word, and the resulting phonemic representations produced by each rule are concatenated to produce a phonemic representation for the morph. For example, in translating "gate", the marker first points to "g". After attempting to apply a number of rules pertaining to the pronunciation of "g", the algorithm reaches the last rule which states that "g" is pronounced "g" (where the latter "g" is an IPA symbol). The pointer then moves on the "a" and the algorithm finds a rule stating that "a" followed by a single consonant, followed by an "e", "i", or "y" is pronounced "e" (again in IPA notation). The pointer moves to "t" which is determined to be pronounced "t". Similarly, "e" is determined to be silent, and the resulting translation of the word is "g" + "e" + "t".

The rules are not independent; rather they are listed in the order to be applied; and thus each rule assumes that no rule listed before it has applied.

Chapter IV

Sentence Parsing

1. Overview

Each sentence of the input stream is parsed on the basis of the part-of-speech information. Only noun and prepositional phrases are detected, resulting in a partial parse. This partial surface structure is used to determine a partial over-all stress contour for the sentence. A complete parse is not attempted since even if SIMON were given parts-of-speech information on every word in the sentence a full syntactic parse is not yet a practical automata task in the present state of the art of computer science. The system does try to parse noun and prepositional phrases, since they are relatively easily detected, as does the system by Prof. Jon Allen at MIT which has access to more complete and reliable parts-of-speech information. Psychological studies show that even when only a partial stress contour is provided, a listener will supply the rest, i.e. he will hear the stress contour based on his own understanding of the sentence. Thus, a partial stress contour gives the illusion that it is complete.

The parsing algorithm is necessarily not like most natural language parsing techniques, since SIMON needs to deal with partial knowledge about the function of the words in a sentence. The frequency of unresolvable ambiguities in the sentence structure is higher, as there may be many legal parses possible within the constraints of SIMON's partial knowledge.

The paring algorithm makes a single pass over the sentence to determine possible phrase groupings. The final phrase bracketing is made on the basis of sentential and agreement tests.

2. Determining Possible Phrases

The initial pass over the input sentence can be viewed as a finite state process. A pointer scans across the sentence searching for phrase openers such as articles and prepositions. The parser then attempts to fit each succeeding word into the phrase structure, marking each word which could possibly be the end of the phrase being assembled. When the pointer reaches a word which couldn't possibly be contained in the phrase (such as a verb or a noun which lacks agreement with the rest of the phrase), the parser returns to the state of looking for phrase openers, backing up to the the word following the last phrase ending. Upon completion of the pass, a list of phrase openers, each with a list of valid phrase endings, has been generated.

For example, we might have an input sentence with the associated knowledge about the parts-of-speech:

The	*big*	*bad*	*wolf*	*kills.*
article	*unknown*	*adjective*	*s.noun/pl.verb*	*pl.noun/s.verb*

"The" is marked as phrase opener and the state of the scan is now that of attempting to fit each succeeding word into a noun phrase. "Big" is an unknown word and as such is marked as a possible phrase ending; however, "big" might also be an adjective, so the scan continues trying to put the next words into a noun phrase structure. "Bad" is an adjective and hence fits into the noun phrase mold. "Wolf" has been determined to be a singular noun or a plural verb. Taking it to be a singular noun allows us to mark it as a possible ending for the noun phrase beginning with "the". The form of a noun phrase used in SIMON does not allow nouns to be embedded in a noun phrase, so the parse of the noun phrase is complete. (See Section IV of this chapter for justification of only allowing nouns at the end of noun phrases.) The parser now looks for another phrase as in the initial state of the parse. "Kills" is marked as a potential noun phrase and the parse is complete. The result is the detection of a possible noun phrase starting with "the" and ending with either "big" or "wolf", and a possible noun phrase consisting of "kills".

3. Final Bracketing of Phrases

The criteria for determining the final bracketing of phrases are verb existence and plurality agreement. That is, each sentence must contain a verb which agrees in plurality with a noun phrase. For example, given the input stream:

These	*ripe*	*red*	*apples*	*taste*	*delicious.*
pl.demonstrative	*adjective*	*unknown*	*unknown*	*s.noun/pl.verb*	*adverb*

SIMON,~~by~~ application of the first pass, marks that there is a noun phrase beginning with "these" and ending with either "red", "apples", or "taste". It then assumes that the phrase ending for each phrase makes that phrase as long as possible, i.e., the last of the possible phrase ending for each phrase is assumed. In this case, the phrase beginning with "these" has three possible phrase endings and the last of these, "taste", is assumed initially. However, this bracketing leaves no verb in the sentence, and SIMON therefore now assumes that "apples" is the phrase ending. "Taste" may now function as a plural verb, satisfying the requirement that a sentence contain a verb. Since the noun phrase has been determined to be plural, and "taste" is a plural verb, the plurality agreement criterion is satisfied.

Upon final bracketing of phrase, the part-of-speech of each word is altered to accommodate its fitting into the phrase. For example, "apples" above would be marked as a plural noun, and similarly "taste" would be marked as a plural verb.

4. Note on Accuracy

The algorithm presented here does not always arrive at a correct parse, but rather it arrives at a valid parse within the confines of its limited knowledge of the function of each word in the sentence. The structure for a noun phrase requires a noun ending, which is not grammatically always the case - consider "can opener" where "can" is a noun. The sentence will be erroneously marked with a phrase ending with "can" instead of "opener". However, this error should not greatly affect the stress placement of the entire sentence, since whether we take "can" to be the end of a noun phrase, or we take "opener" to be the end of the phrase, "can" receives primary stress. Hence, detectable errors in stress placement should be minimized.

Chapter V

Stress Placement Within the Word

1. Overview

The main difference between SIMON and previous speech systems not employing a large dictionary of stored words is that SIMON put inflection into its voice output rather than producing monotone speech such as in "Automatic Translation of English Test to Phonetics by Means of Letter-to-Sound Rules", [Elovitz, et al, 1976]. This assignment of inflection entails computing stress contours over phrases as well as determining stress contours over phrases as well as determining stress within each word. Thus, e.g., the system needs to determine that "table" has primary stress on its first syllable. Higher stress on a syllable in this system means that the output of the syllable will be voiced with higher amplitude (i.e. louder), with higher pitch, and for a longer duration of time.

The method of assigning stress levels within the word is the application of the Main Stress Rule and the Alternating Stress Rule as outlined in *The Sound Patterns of English*, [Chomsky and Halle, 1968]. These rules operate on the phonemic representation of each word as determined by a letter-to-sound conversion scheme, and use the morphological information provided by the parsing of the word.

2. Application of Stress Rules

The Main Stress Rule is based on the placement of strong and weak clusters of phonemes. A weak cluster is a simple vocalic nucleus followed by no more that one consonant, followed by an optional "r" or "w". A strong cluster is simply one which is not weak. A simple vocalic nucleus is one such as in "gut", "get", or "got", whereas a complex vocalic nucleus is one such as in "goat" or "gate". For example, "complete" has two strong clusters, and "adaptation" has a weak cluster, followed by two strong clusters and another weak cluster. The Main Stress Rule in its simplest form states that primary stress falls on the penultimate cluster of a word if the last cluster is weak, and on the last cluster otherwise. Hence, "complete" receives primary stress on the last syllable, and "adaptation" receives primary stress on the second to last syllable.

The Alternating Stress Rule is also used and accounts for the stress placement in words such as "candidate" which receives primary stress on its last syllable by application of the Main Stress Rule. The Alternating Stress Rule states that when the Main Stress Rule has assigned primary stress to the last syllable, the third to the last syllable should receive primary stress, reducing the last syllable to secondary stress. Thus, "candidate" receives primary stress on the last syllable by the Main Stress Rule and then the Alternating Stress Rule shifts primary stress to the first syllable, giving the last syllable secondary stress to arrive at the correct stress placement.

The Main Stress Rule is actually a bit more complicated than the rule as I have so far stated it here. However, for simplicity of explanation, I will continue to refer the Main Stress Rule as the rule stated above, and will impose certain conditions on its use. The Main Stress Rule as given by Chomsky incorporates these conditions into the Main Stress Rule. We ignore the last syllable before applying the Main Stress Rule if either of the following conditions exist:

(1) the word is a noun or adjective and the last syllable is a derivational suffix with a lax vowel.

(2) the word is a noun and the last syllable is a lax vowel followed by zero or more consonants.

For example, "personal" has the derivational suffix "-al". Applying the Main Stress Rule to the root left over after taking the suffix away as per condition (1) above we put primary stress on the first syllable. Condition (2) applies to "asterisk"; therefore, we ignore the last syllable and apply the Main Stress Rule to "aster" which receives primary stress on the first syllable, indicating that "asterisk" receives stress on the first syllable.

Chapter VI

Stress Contours For Phrases

1. Overview

The grouping of phrasal units in the sentence allows the assignment of stress contours for the sentence. A stress contour of a phrase gives the relative inflection of each word in the phrase. For example, the phrase "the bad wolf" has a rising stress contour, i.e. each word receives an increasing degree of stress. "The big bad wolf" has a stress contour of (4, 2, 3, 1), meaning that "the" receives the least stress (a higher number corresponds to less stress), "wolf" receives primary stress, while "big" and "bad" receive a stress level between these two, with "big" having a higher stress than "bad".

There are reliable rules for determining the stress contour given the structure of the phrase as given by *The Sound Patterns of English*, [Chomsky and Halle, 1968]. SIMON assumes that an adverb modifies an adjective to its right and is left associative, and that an adjective modifies a noun to its right and is right associative. Thus a noun phrase of the form

$(adv_{11}, \ldots, adv_{1m}, adj_1, adv_{21}, \ldots, adv_{2m}, adj_2, \ldots, noun)$

is grouped

$(((adv_{11} adv_{12}) adv_{13} \ldots) adj_1, (((adv_{21} adv_{22}) adv_{23} \ldots) adj_2 \ldots \mathbf{noun}))$

The Nuclear Stress Rule is applied successively until a stress contour for the phrase is determined.

2. Stress Assignment

The system assigns stress according to an algorithm equivalent to application of the Nuclear Stress Rule. The noun is given primary stress and articles or demonstratives are given the least stress. A left-to-right pass is made over the phrase assigning secondary stress to the first adjective, tertiary stress to the second adjective, and so on. A right-to-left pass is then made over the phrase assigning successively lower stress levels to the adverbs associated with each adjective; e.g., if the if the adjective has secondary stress, then the associated adverbs receive tertiary stress, quaternary stress, and so on, marking these adverbs right-to-left.

For example, "the big bad wolf" initially gets marked with stress values of

```
The  big  bad  wolf
 |               |
 4               1
```

The first pass is then made over the phrase assigning stress values to the adjectives arriving at

```
The  big  bad  wolf.
 |    |    |     |
 4    2    3     1
```

The second pass does not do anything in this case since there are no adverbs.

"The very big terribly bad wolf" initially gets marked

```
The  very  big  terribly  bad  wolf.
 |                               |
 4                               1
```

The *first pass yields*

```
The  very  big  terribly  bad  wolf.
 |          |              |     |
 4          2              3     1
```

The second pass now assigns stress to the adjectives giving

The	*very*	*big*	*terribly*	*bad*	*wolf.*
|	|	|	|	|	|
4	3	2	4	3	1

Hence, formal bracketing is not required since the algorithm uses the implicit structure given by the ordering of adjectives and adverbs.

3. The Compound Rule

The Compound Rule as given in *The Sound Patterns of English*, [Chomsky and Halle, 1968] is also applicable inside of phrases and assigns primary stress to the leftmost word of a compound structure; e.g., the rule assigns primary stress to "can" in "can opener" where both "can" and "opener" function as nouns. However, this rule is not used in SIMON's stress assignment algorithm because the phrase parser does not allow compound structures, i.e. all phrases must end with a noun. This erroneous assumption in parsing does not make a significant difference in stress assignment. Consider "The very new can opener is..." Proper parsing and application of the Nuclear Stress Rule and the Compound Rule such as in Prof. Jon Allen's system at MIT yields the contour

4 3 2 1 4

whereas SIMON's parsing and stress assignment technique yields:

4 3 2 1 -

and leaves "opener" with neutral stress. The justification for SIMON's methods is seen to be quicker, less complicated parsing and stress assignment algorithms at the cost of not recognizing the full phrase and a slightly modifies stress contour.

Chapter VII

Final Adjustments For Voice Output

After the system has arrived at an inflected phonemic representation for the sentence, minor modifications are needed for more intelligible speech output. Short pauses are placed before each phrase, after commas and semicolons, and before each word that open a clause as per recommendations of "Machine-to-man communication by speech Part II: Synthesis of prosodic features of speech by rule", [Jonathan Allen, 1968]. Also in accordance with the MIT's system, longer pauses replace colons, and ends of sentences. Sentences ending with a question mark receive a rising stress contour at the very end of the sentence unless the sentence begins with "who", "where", "why", "how", or "when". Finally, the IPA phonemes re translated into phonemes used by the VOTRAX ML-I Voice Synthesizer for output. The correspondence is close to one-to-one, however, insertion of a short vowel is occasionally needed. For example, an "l" starting a word needs an "eh" (the "e" of "get") following it to produce more natural speech

Chapter VIII

Suggestions For Further Work

A stricter adherence to Chomsky's rules of English phonetics would help a great deal in enhancing the quality of speech. The rules could be incorporated in data driven routines for interpreting the rules, rather than having the rules procedurally embedded as in the SIMON voice synthesis system. To make real time applications more realistic, the methods described in this thesis could be implemented with special purpose micro-processor configuration, and/or a scheme for parallel processing of the computations involved. A better memory management system is needed in SIMON as the system does not allow for shared data structures.

Appendix I

Implementation

SIMON has been implemented in a recursive FORTRAN on UYK-20 under the SHARE-7 time-sharing system. The speech is output through a VOTRAX ML-I Voice Synthesizer. The implementation uses data-structures with dynamic allocation of storage by means of a pointer system. Deallocation of storage is done "manually", i.e. each routine is responsible for deallocating unused memory. The program is very modular and aims at clear code rather than conserving space or making the routines run as fast as possible.

Appendix I

Memory Management

Allocation of memory is achieved by creating a linked list of unused cells in memory, i.e. each unused cell contains a pointer to the location of another unused cell. When a data-structure needs memory, a cell is taken out of the linked list of unused cells, and a pointer to the freed cell is given to the data-structure. Memory is deallocated by inserting a cell into the linked list of free storage.

Appendix I

Data Structures

The data structures used in SIMON are:

1) STORAGE - structure for unused storage, allows allocation and deallocation of memory

2) STRING - a string of ASCII characters

3) IPA - a string of phonemes and associated inflection levels

4) MORPH - a structure containing the graphemic and phonemic representation of a morph, along with the type of morph

5) TREE - a general purpose tree structure of pointers to other data-structures

6) SENT - a structure for sentences containing an ordered list of words and punctuation marks

7) PSEN - a structure for parsed phrases containing an ordered list of WORDs and PSENs

8) WORD - a structure for words containing the phonemic and graphemic representations, and the part-of-speech of the word along with a list if composite MORPHs

9) LEX - a lexicon containing some irregular morphs, all affixes, and contains the type of each morph. In the case of suffixes it contains the part-of-speech produced by adding the suffix to a morph, and the allowable morphs to which the suffix may be added.

All the data-structures are achieved by dividing up each 32-bit word into two 16-bit fields. Each data-structure knows whether to interpret the fields as data or as pointers. In addition to the information contained in the data-structures as listed above is an internal piece of information telling the type of data-structure; this information prevents a pointer to one type of data-structure as being mistaken as a pointer to another type of data-structure.

Appendix I

Program Structure And Data Flow

The procedural modules used in SIMON are:

1) MAIN - coordinates the flow of control and data of the other modules

2) GETSEN - gets a sentence from the input text file

3) PARSE - parses a sentence into phrases composed of parsed words

4) PARWOR - parses a word into its constituent morphs

5) TRANS - applies letter-to-sound rules to morphs to arrive at a phonemic representation for the morph

6) MATCH - determines whether a string of characters matches a set of pattern- recognition symbols

7) COMBIN - applies rules for stress placement to a parsed sentence

8) VOTRAX - inserts pauses, adjusts inflection for questions, and prepares output for the VOTRAX voice synthesizer

9) MUMBLE - sends voice parameters to VOTRAX output channel

The MAIN routine gets a SENT from GETSEN. This SENT is passed to PARSE which sends each STRING in the SENT to PARWOR. PARWOR breaks the STRING into constituent morphs and sends the graphemic representation of the morph (a STRING) to TRANS which returns the phonemic representation of the morph (an IPA) to PARWOR which returns a parsed word (WORD) to PARSE. Phrases are parsed in PARSE and a PSEN is returned to MAIN which gives it to COMBIN. Stress levels are computed in COMBIN and an IPA is returned to MAIN for final adjustments, sending the IPA to MUMBLE for vice output.

References

Allen, Jonathan, "Machine-to-man communication by speech, Part 2: Synthesis of prosodic features of speech by rule," *1968 Spring Computer Conference, AFIPS Proceedings*, vol. 32, Washington, D.C.: Thompson, 1968, pp 339-344.

Chomsky, N. and Halle, M. *Sound Patterns of English*, New York: Harper and Row, 1968

Elovitz, H.S. et al, "Automatic Translation of English Text to Phonetics by Means of Letter-to-Sound Rules", Naval Research Laboratory, Report 7948, 1976

Holmes, J.N. "Speech Synthesis by Rule," *Language and Speech*, vol.7, pt. 3, pp. 127-143, July-September, 1964

Knuth, Donald, "Preliminary description of TEX", 1977

Kucera, H., *Computational Analysis of Present-Day American English*, Brown University Press, Providence, 1967

Lee, F. F., "Machine-to-man communication by speech, Part 1: Generation of segmental phonemes from text," *1968 Spring Computer Conference, AFIPS Proceedings*, vol. 32, Washington, D.C.: Thompson, 1968, pp 333-338.

McIlroy, M.D., "Synthetic English Speech by Rule", Bell Telephone Labs, Inc., 1974

Winograd, Terry, "An AI Approach to English Morphemic Analysis", MIT AI Lab, Memo 241, February, 1971

INITIAL DISTRIBUTION LIST

Addressee	No. of Copies
ONR, ONR-240 (Joel Trimble)	1
CNM, MAT-08T1 (CAPT Parrish)	1
NAVPGSCOL	1
DTIC	12

DAT

FILM

4 —